Path Segment Identifier (PSID) in SRv6: The Flag Problem Options Analysis

- Document: draft-ietf-spring-srv6-path-segment
- Presenter: Guanming Zeng, Huawei Technologies
- IETF125 SPRING Working Group Meeting

Background & Objectives

Problem:

- SRv6 lacks a unified Path Identifier comparable to SR-MPLS PSID (RFC 9545), using the variable-length Segment List as a key is inefficient
- Goal: Define an efficient, unique 128-bit PSID (path segment identifier) mechanism to support:
  - Bidirectional Path Binding
  - End-to-End Path Protection (1+1)
  - Performance Measurement (PM)

Protocol Design:

- **Location**: PSID must reside at SegmentList[n] where n = SRH.LastEntry.
- **Structure**: Follows standard SRv6 SID format (Locator:Function:Argument), where Function = END.PSID (0x0064).
- **The flag problem: How to efficiently signal to nodes that this specific entry is a Path Identifier and not a standard routing SID?**

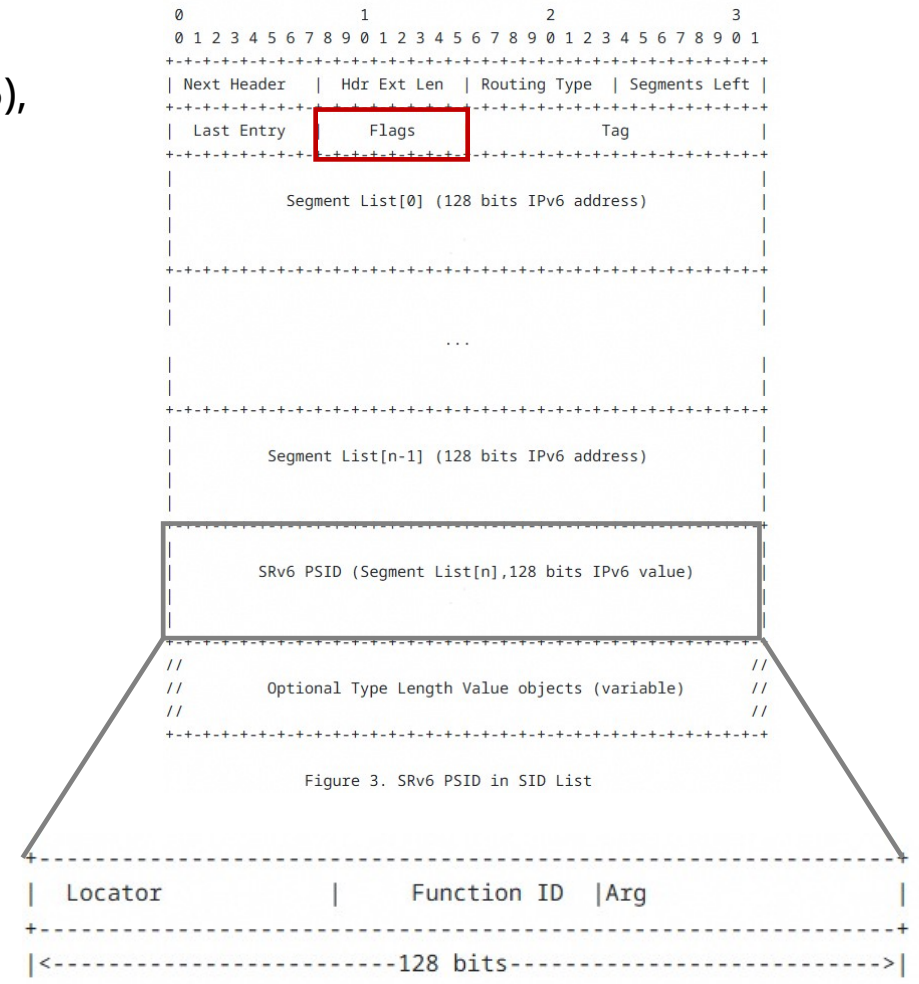


Figure 3. SRv6 PSID in SID List

Figure 2. PSID Format following LOC:FUNCT:ARG

Overview of Proposed Options

We present three high-level approaches for community review:

Option 1: Dedicated P-flag (Two Sub-options)

Dedicated P-flag specifically for PSID/ P-flag for generic data

Option 2: Generic G-Flag

Introduce a generic flag indicating the last entry is a structured 128-bit sid

Option 3: No New Flag (Three Sub-options)

Reuse O-flag / Pure SID Convention / Dedicated Prefix Matching.

Option 1A – Dedicated P-flag

Mechanism:

- Define a new bit in the SRH Flags field: P-flag.
- Semantics: P=1 explicitly indicates that SegmentList[Last Entry] carries a PSID.

Pros:

- Simple & Unambiguous: Direct logic, easy to implement.

Cons:

- Resource Consumption: Consumes 1 of only 8 available SRH flag bits for a single function.
- Limited Extensibility: Specific to PSID; does not accommodate future metadata types.

Noting: Option 1A is already realized by ZTE and H3C.

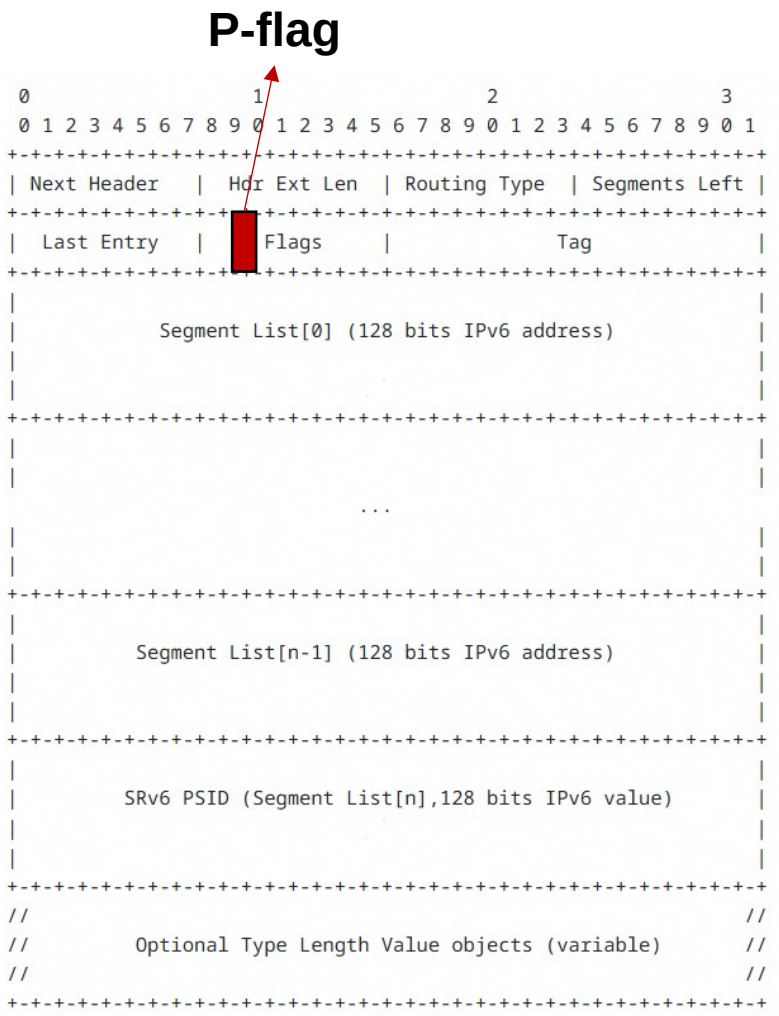


Figure 3. SRv6 PSID in SID List

Option 1B – P-flag for Generic data

Mechanism:

- Define a new bit in the SRH Flags field: P-flag.
- Semantics: P=1 explicitly indicates that SegmentList[Last Entry] carries a **128-bit generic data**. PSID is one of the possible generic data.

Pros:

- Future Extensibility: One flag supports multiple extensions, addressing the "flag scarcity" concern.
- Compatible with existing implementation of option 1A.

Cons:

- Resource Consumption: Consumes 1 of only 8 available SRH flag bits.

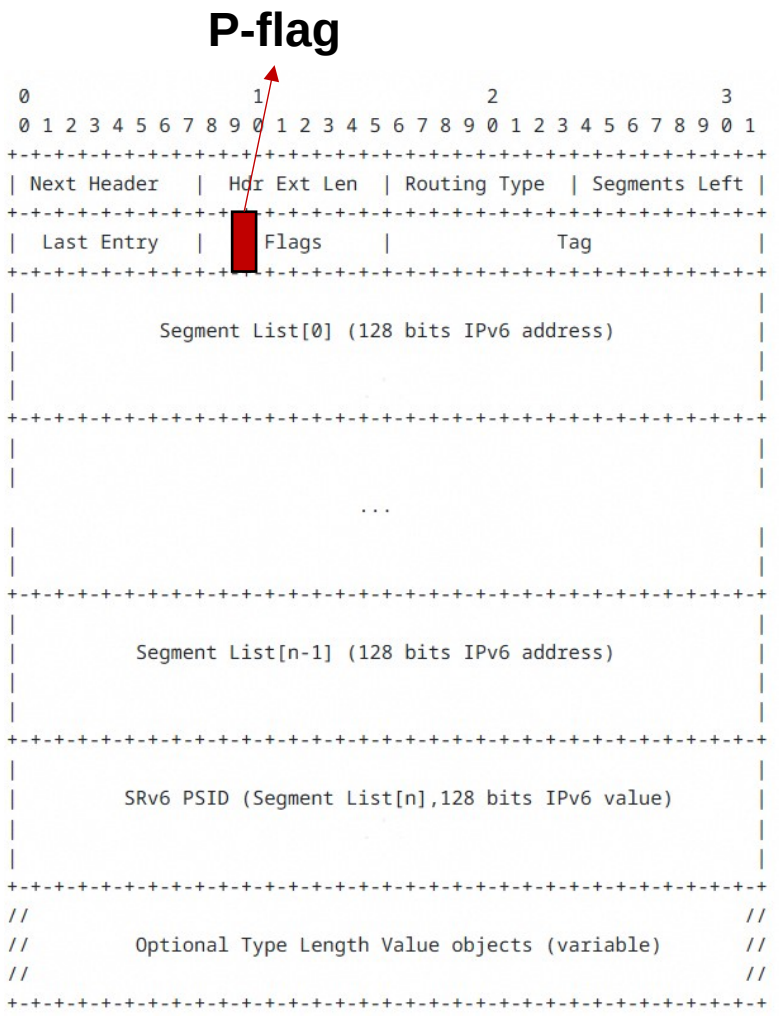


Figure 3. SRv6 PSID in SID List

Option 2 – Generic Flag

Mechanism:

- Define a generic flag bit: G-flag (Generic).
- Semantics: G=1 signals that SegmentList[Last Entry] contains a 128-bit data with SID structure. The opcode is defined to distinguish different use cases, for example:
  - OpCode=0x01: Path Segment ID (PSID)
  - OpCode=0x02: In-situ OAM trace data
  - OpCode=0x03: ....

Pros:

- Future Extensibility: One flag supports multiple extensions, addressing the "flag scarcity" concern.

Cons:

- Resource Consumption: Consumes 1 of only 8 available SRH flag bits for a single function.
- SID structure is needed to be defined.

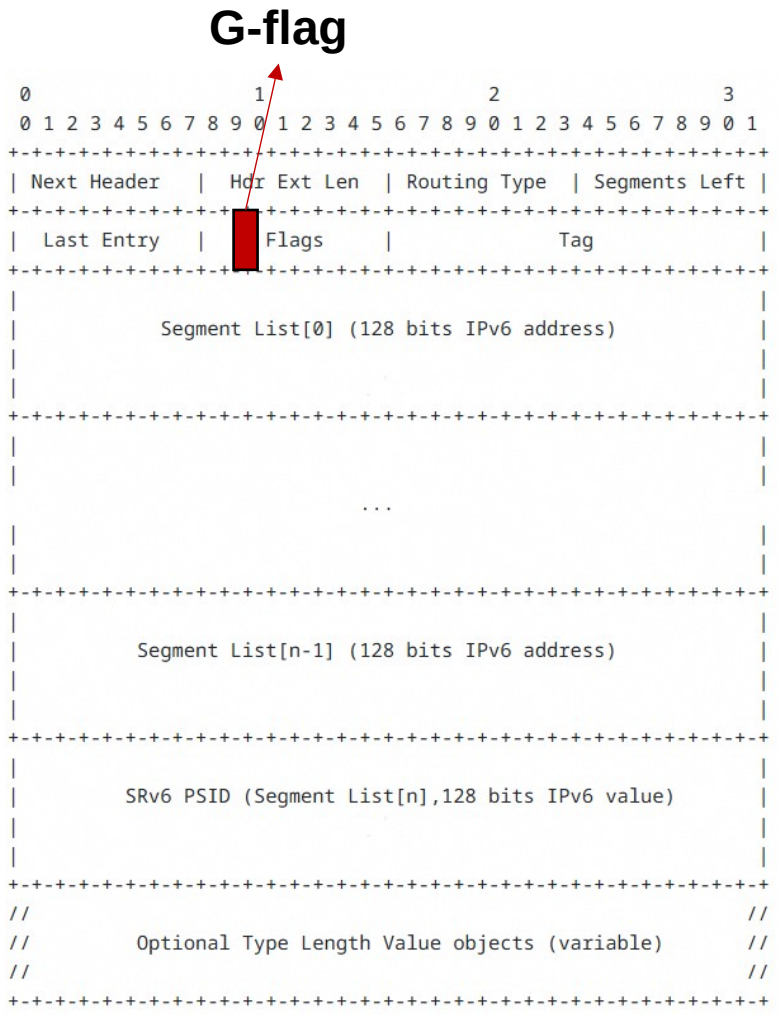


Figure 3. SRv6 PSID in SID List

Option 3 – No New Flag (Sub-options)

3A: Reuse O-flag (OAM Flag)

Mechanism:

- Use the existing OAM flag to signal PSID presence （RFC 9259）.

Pros:

- No SRH flags consumption.

Cons:

- O-flag implies slow-path, sampled OAM treatment, but PSID often requires fast-path, per-packet handling for accurate end-to-end metrics. Mismatch in processing model risks under-serving key use cases.

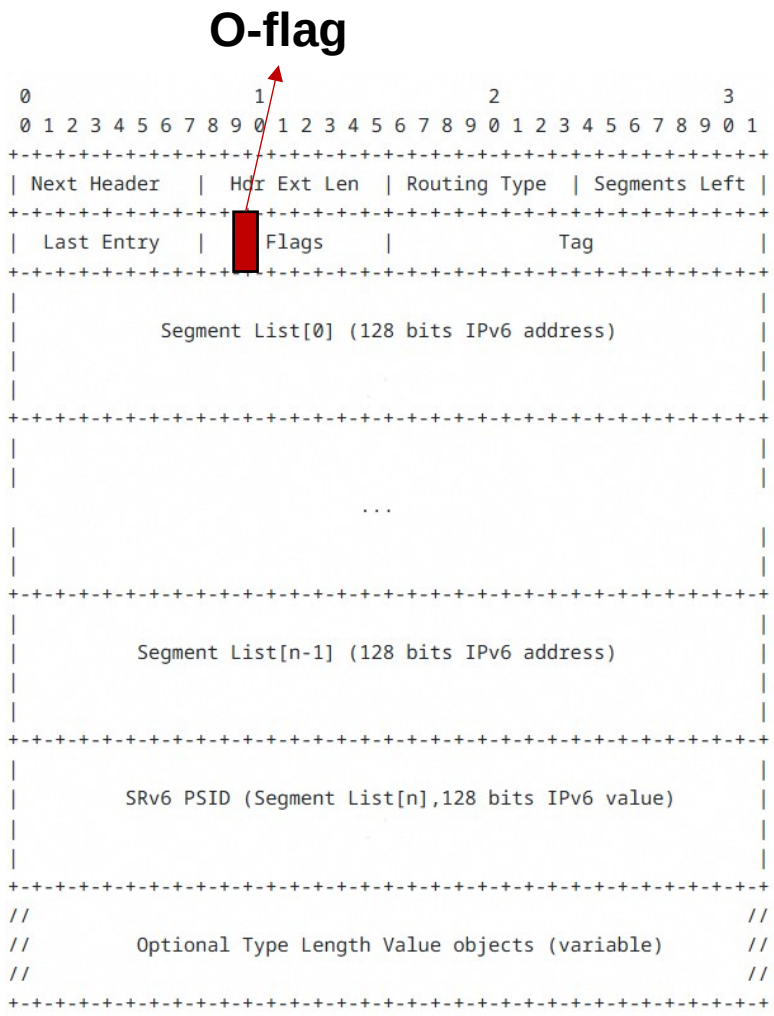


Figure 3. SRv6 PSID in SID List

Option 3 – No New Flag (Sub-options)

3B: Flag-less (Pure SID Convention)

Mechanism:

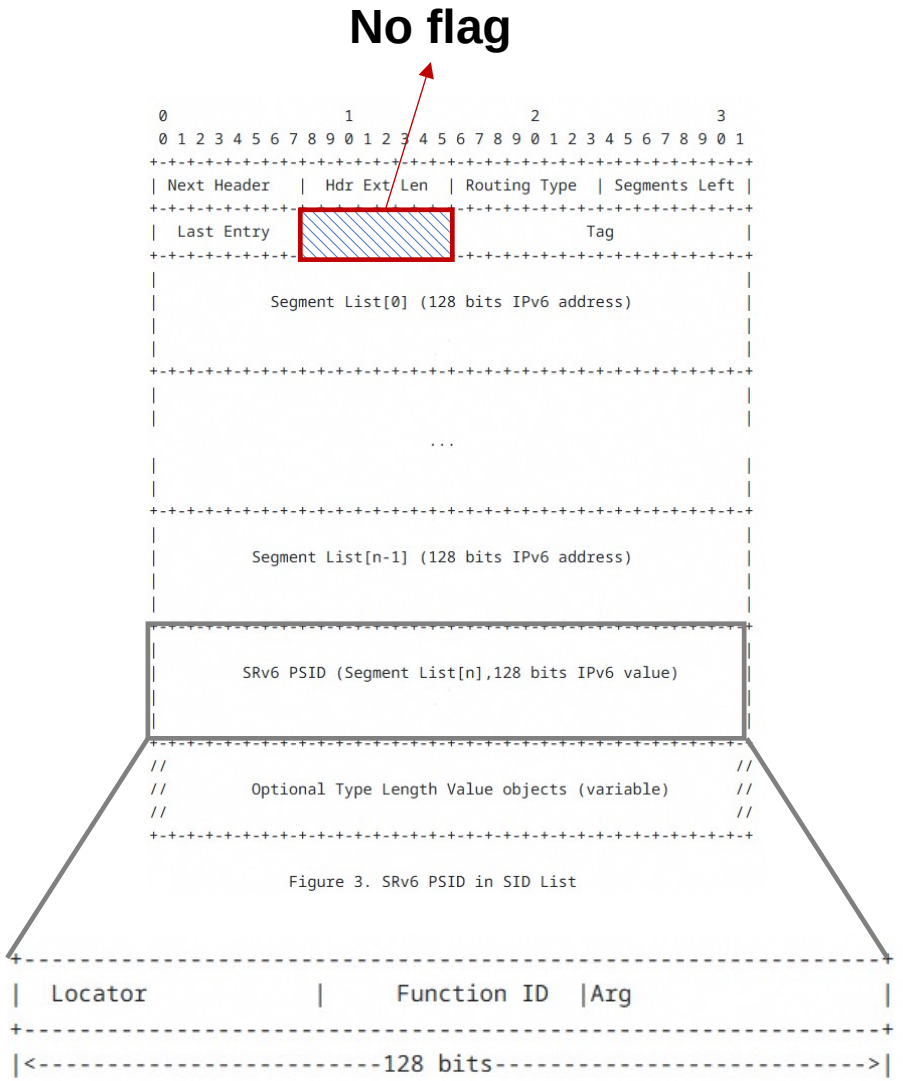
- SR Endpoint nodes inspect SegmentList[n] by convention without defining a flag.

Pros:

- No SRH flags consumption.

Cons:

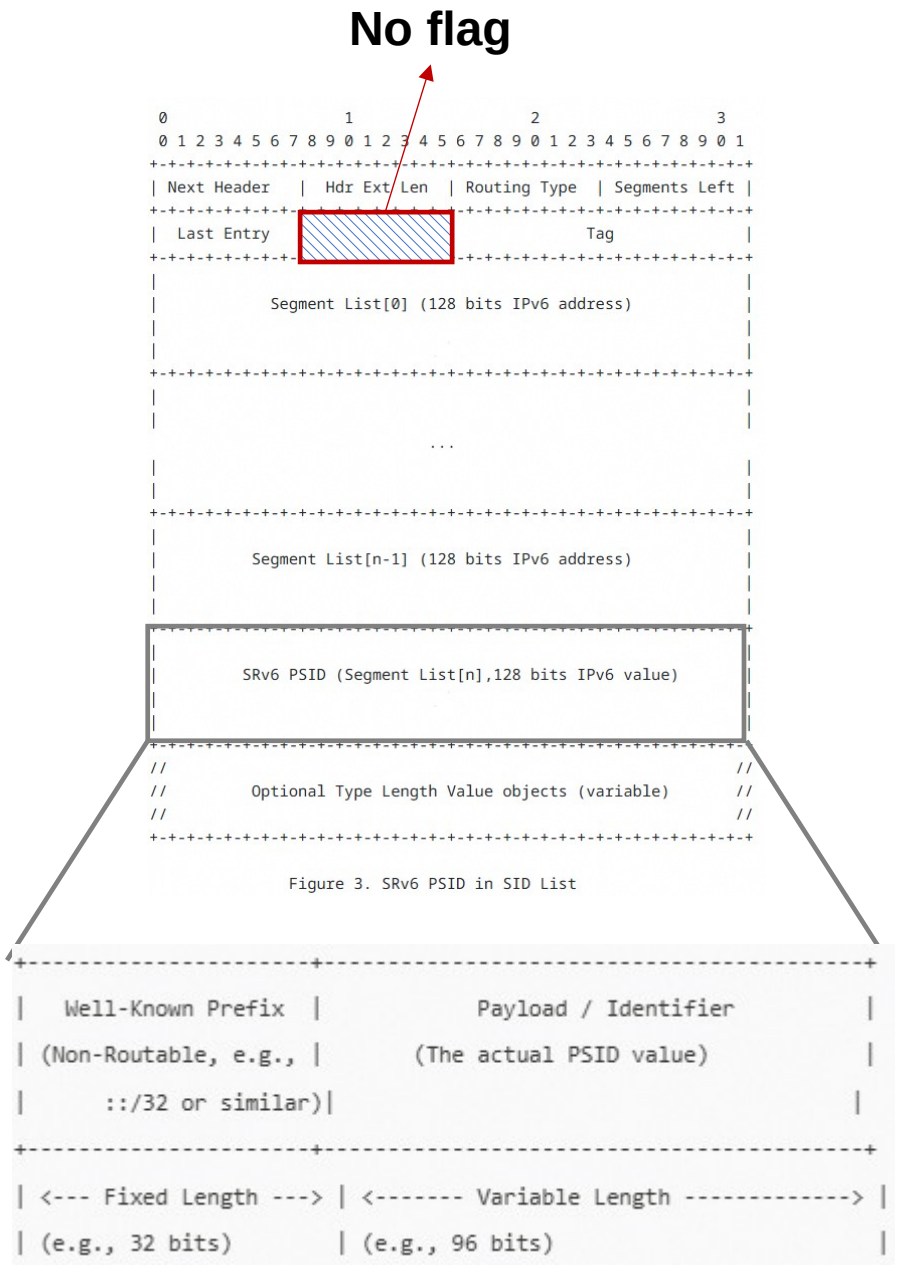
- Without a flag, SR nodes on the path needs reading the 128-bit PSID, which introduces more complexity than just reading one flag
- Without a flag indicating legal data on segment list[n], the packet may be treated as illegal and discarded by strict implementations



Option 3 – No New Flag (Sub-options)

3C: Flag-less with Dedicated PSID Prefix

- **Mechanism**:
  - SR Endpoint nodes inspect SegmentList[n] and recognize PSID by prefix match.
  - Reserve a well-known, non-routable IPv6 prefix (e.g., ::/32) for PSIDs.
- **Pros:**
  - No SRH flag consumption.
- **Cons:**
  - Without a flag, SR nodes on the path needs reading the 128-bit PSID, which introduces more complexity than just reading one flag
  - Without a flag indicating legal data on segment list[n], the packet may be treated as illegal and discarded by strict implementations
  - Extra mechanism is needed to ensure the PSIDs are unique across domains because of not using the egress node's Locator
  - Extra mechanism is needed to inform which SR Endpoint Node triggers corresponding operation



Comparative Analysis Matrix And Call for Consensus

Feature	Opt 1A (P-flag)	Opt 1B (P-flag+generic data)	Opt 2 (G-flag)	Opt 3A (O-flag)	Opt 3B (Pure SID)	Opt 3C (Prefix)
Flag Consumption	1 bit	1 bit	1 bit	0 (Reuse)	0	0
Processing Model	Fast	Fast	Fast	Slow/Sampled	Fast	Fast
Extensibility	Low (PSID only)	High	Medium	Low	Low	Low
Processing Cost	Low	Low	Low	Low	High (Reading 128 bit sid)	High (Reading 128 bit sid)
Compatible with existing P-flag implementation	Yes	Yes	No	No	No	No

Authors' Recommendation:

- We believe **Option 1A(P-flag)** is simple, unambiguous, **Option 1B(P-flag+generic data) and option 2 (Generic Flag)** offers the best long-term balance: it conserves scarce flag space, supports future extensions (beyond PSID), and maintains performance.

Call for Consensus:

We kindly ask the WG to share your views on:

- Which direction best meets operational and architectural needs?

Depending on feedback, we will update the draft accordingly.

Thank you!

Note: Use cases

Use Case1– Bidirectional Path Association

Objective:

Bind two unidirectional SRv6 paths (Forward & Reverse) into a single logical session for coordinated management and fast fault notification (e.g., Mobile Backhaul).

The Challenge:

- Forward and Reverse paths often have different SID Lists and traverse different nodes.
- Without a common identifier, endpoints cannot locally associate the two directions for fast signaling.

PSID Workflow:

① Setup: Controller assigns PSID_Fwd (e.g., 0x5555) and PSID_Rev (e.g., 0x6666).

② Binding: Both Ingress/Egress nodes are configured with a mapping table: 0x5555 <--> 0x6666 (Same Session ID).

③ Fault Detection:

- Egress of Forward path detects loss on PSID_Fwd.
- Local Logic: Immediately looks up the bound PSID_Rev.

④ Fast Notification:

- Node triggers an OAM/BFD signal on the Reverse path (PSID_Rev) to alert the far end.
- Action is taken locally in milliseconds, without waiting for controller intervention.

Key Benefit:

- Session Awareness: Turns two independent unidirectional tunnels into a managed bidirectional session.
- Fast Convergence: Enables sub-50ms coordinated protection switching.

Note: Use cases

Use Case 2 – End-to-End 1+1 Path Protection (Live-Live)

Objective:

Implement "Dual-Transmit, Select-Receive" protection where the Egress node accepts packets from the Primary path and discards duplicates from the Secondary path.

The Challenge:

- Primary and Secondary paths have completely different SID Lists.
- The Egress node must recognize that two different incoming packets are redundant copies of the same data to avoid duplication.

PSID Workflow:

1. Configuration: Egress node defines a Protection Group:
    - Primary Member: PSID_A, Secondary Member: PSID_B, Policy: Prefer Primary; Drop Secondary if Primary received.
2. Transmission: Ingress duplicates traffic:
    - Copy 1 → Path A (Tagged PSID_A)
    - Copy 2 → Path B (Tagged PSID_B)
3. Selection Logic (Egress ASIC):
    - Packet Arrives (PSID_A): Matched as Primary → Forward to user. Start short "Wait Window".
    - Packet Arrives (PSID_B): Matched as Secondary. If Primary received in window: Drop (Duplicate). If Primary missing (Timeout): Forward (Failover active).

Key Benefit:

- Zero Duplication: Hardware-level deduplication based on Group ID, not payload inspection.
- Hitless Switchover: Seamless transition during failures with no packet reordering issues.

Note: Use cases

Use Case 3 – Performance Measurement (Passive)

Objective:

Correlate packet counts and timestamps at Ingress/Egress nodes to calculate precise Packet Loss and Delay without impacting forwarding performance.

The Challenge:

- Matching full SID Lists for correlation is computationally expensive for hardware counters.
- Dynamic path changes break statistical continuity if based solely on topology.

PSID Workflow:

1. Ingress Node: Encapsulates traffic with SRH containing a unique PSID (e.g., 0x1001). Records Tx_Count & Tx_Timestamp keyed by PSID.
2. Transit: Packet forwards normally; PSID remains intact in the header.
3. Egress Node: Parses SRH, extracts PSID. Records Rx_Count & Rx_Timestamp keyed by the same PSID.
4. Controller/Analyzer: Receives reports from both ends. Matches records instantly using PSID as the Unique Key.
    - Calculation: Loss = Tx(PSID) - Rx(PSID) | Delay = Ts_Rx(PSID) - Ts_Tx(PSID)

Key Benefit:

- Hardware Efficiency: O(1) lookup time for counters vs. complex list matching.
- Decoupling: Statistics remain consistent even if the underlying SID List changes.

Note: HBH vs PSID

Q： Can HBH header replace PSID ?

- **Performance Mismatch**: In most current implementations, packets carrying HBH options are punted to the **slow path (CPU/software processing).** (**Many existing devices do not support process HBH is fast path**). PSID is designed for **high-frequency, per-packet operations**.
- **Extra processing**: HBH header is processed by each IPv6 node, however PSID is processed by each SRv6 Endpoint node. Using HBH header will introduce unnecessary and extra processing by IPv6 node. For use case like b**idirectional path association, the switching should be triggered as fast as possible.**

Option 3 – No New Flag (Sub-options)

3B: Flag-less (Pure SID Convention)

1. Handling of Unknown Content

In the Flag-less model, the Segment List[n] entry contains a PSID value.

The Risk is: Without a Flag to explicitly mark this entry, strict implementations performing pre-checks on the SRH might attempt to validate the length of the Segment List. And **putting a PSID on Segment List[n] may be treated as illegal and discarded**.

2. Forwarding Plane Efficiency

From a data plane implementation perspective, the presence of a Flag fundamentally optimizes the processing pipeline.

•**Current Flag-less Flow**: Generally, when receiving a packet, an SR node processes according to these steps:

1.Read the SRH Flags field.

2.Read and Parse Segment List[n] (regardless of whether it will be used immediately).

Issue: The node is forced to fetch and potentially validate the metadata value even if it's not the active segment, **adding unnecessary memory access and logic overhead**.

•**Proposed Flag-based Flow**: If a G-flag is defined, the processing logic becomes more efficient:

1.Read the SRH Flags field.

2.Check the G-flag:

If Unset: The node knows Segment List[n] does not contain a routing SID. **It skips reading/parsing that specific entry entirely for forwarding decisions**.

If Set: The node proceeds to process the entry as a PSID or other meta data.

Benefit: This allows the hardware to bypass unnecessary memory reads and validation logic for metadata entries.

Note: Segment List ID vs PSID

SLI: The "Internal Order Number" (Control Plane Only)

- Source: draft-ietf-idr-sr-policy-seglist-id
- Where it lives: **Inside BGP Update Messages** (Sub-TLV).
- Who sees it: **Only the Controller and the Headend Node**.
- Lifecycle:
  - Controller sends Policy + SLI to Headend.
  - Headend stores mapping: SLI 100 <-> Segment List {A, B, C}.
  - Packet is generated: The SLI is NOT inserted into the packet header.
- Primary Use Case:
  - Efficient reporting: Headend sends stats back to Controller using short ID instead of long SID list.
- Limitation:
  - Zero Visibility beyond the Headend. Intermediate nodes are blind to the SLI.

PSID: The "Shipping Label" (Data Plane Everywhere)

- Where it lives: **Inside the IPv6 Packet Header** (SRH Flags/Fields).
- Who sees it: **Every SR-aware node along the path (Headend, Transit, Tail)**.
- Lifecycle:
  - Headend encodes PSID into the packet.
  - Packet travels across the network.
  - Every hop reads the PSID to identify the logical path.
- **Primary Use Cases**:
  - Mid-path Telemetry: Transit nodes insert timestamps/metrics based on Path ID.
  - Fast Reroute (FRR): Protecting nodes switch traffic based on Path ID instantly.
  - Path Differentiation: Distinguish overlapping paths with different service levels.